Virtual Triggering: a Technique to Segment Cryptographic Processes in Side-Channel Traces

Jeremy Guillaume, Maxime Pelcat, Amor Nafkha, Rubén Salvador

Virtual Triggering: a Technique to Segment Cryptographic Processes in Side-Channel Traces

1st Jeremy Guillaume
IETR UMR CNRS 6164
CentraleSupélec Rennes Campus
35576 Cesson-Sevigné, France
jeremy.guillaume@centralesupelec.fr

2nd Maxime Pelcat
IETR UMR CNRS 6164
INSA Rennes
35700 Rennes, France
maxime.pelcat@insa-rennes.fr

3rd Amor Nafkha
IETR UMR CNRS 6164
CentraleSupélec Rennes Campus
35576 Cesson-Sevigné, France
amor.nafkha@centralesupelec.fr

4th Rubén Salvador
IETR UMR CNRS 6164
CentraleSupélec Rennes Campus
35576 Cesson-Sevigné, France
ruben.salvador@centralesupelec.fr

***Abstract*—Side-Channel Attacks (SCAs) exploit data correlation in signals leaked from devices to jeopardize confidentiality. Locating and synchronizing segments of interest in traces from Cryptographic Processes (CPs) is a key step of the attack. The most common method consists in generating a trigger signal to indicate to the attacker the start of a CP. This paper proposes a method called Virtual Triggering (VT) that removes the need for the trigger signal and automates trace segmentation. When the time between repetitions is not constant, further trace alignment techniques are required. Building on VT, we propose a simple method to learn representative segment templates from a profiling device similar to the victim, and to automatically locate and pull out these segments from other victim devices using simple pattern recognition. We evaluate VT on screaming channel attacks [1], which initially used a Frequency Component (FC) known to appear at a single time in leaked signals, as a trigger to segment traces. We demonstrate that VT not only performs equivalently to FC on a standard attack scenario, but we also show how using VT with the automatic pullout technique improves the attack efficiency and enables more realistic attack scenarios. Thanks to VT, screaming channel attacks can now: (1) succeed with only half of the segments collected compared to the FC trigger from the original attack; and (2) absorb time variations between CPs.**

***Index Terms*—Cybersecurity, side-channel attacks, screaming channels, electromagnetic side-channels, trace collection.**

I. Introduction

Side-Channel Attacks (SCAs) [2] exploit data-correlated leakages that arise when a device operates on data. The term *side-channel* is used to denote physical leakage signals carrying confidential information. Side-channels are inherent to CMOS computing devices and can take many forms, from timing to power consumption to Electromagnetic (EM) emanations. We call *traces* to the sampled measurements of a side-channel during the execution of one or multiple targeted operations. The most common scenario in SCAs targets a cryptographic key manipulated by Cryptographic Processes (CPs), as eavesdropping such a key can jeopardize system confidentiality. We refer to these attacks as Side-Channel Cryptographic Attacks (SCCAs) [3]. A trace can contain leakage from multiple CPs. Trace segmentation separates the different segments from the leakage, each corresponding to the side-channel measurement of one CP execution. During the attacking phase, a hypothesis $\hat{y} \in Y$ on the data y is made (Y = all possible data values). Techniques like Differential Power Analysis (DPA) [4], Correlation Power Analysis (CPA) [5], Mutual Information Analysis (MIA) [6], template attacks [7] and more recently Deep Learning (DL) [8], evaluate the relationship between $\hat{y}$ and the segment leakage values. Each hypothesis Y is tested, and a probability is returned for all of them. The $\hat{y}$ having the highest probability is expected to correspond to the data y. For these attacks to work, it is important to know, for each segment point, to which CP operations they belong. It makes it possible to find a relation between y and leak values of the same CP data-correlated operations present in all segments. To respect this requirement, segment synchronization is done during the collecting and pre-processing phases of the attack.

Focused on screaming channel attacks, we look into its collecting and pre-preprocessing phases, which aim at obtaining synchronized and denoised segments from the victim's leakage traces using techniques like time diversity (average multiple CPs computing the same data to reduce the noise). To relax the triggering requirements for trace capturing and synchronization, our contributions include:

- *Virtual Triggering (VT)*, a method to segment traces from side-channels of an embedded device executing a cryptographic software implementation. VT requires neither external synchronization nor tampered victim software.
- an experimental evaluation of the proposal on a realistic screaming channel attack to Advanced Encryption Stan-

978-1-6654-8524-1/22/$31.00 ©2022 IEEE

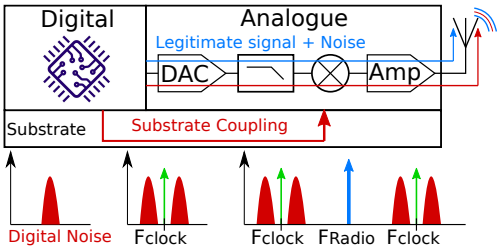

Fig. 1. **Screaming channel attacks:** Conventional side-channels leak to the RF module in the analog part, present on the same die. This one transmits side-channels at a larger distance (until some meters).

dard (AES) on an embedded device.

- a discussion on the method limitations and solutions.
- experimental results demonstrating the gains obtained with these solutions.

VT does not require any specific setup on the victim side, like a trigger signal to indicate the start of a CP. It consists in finding a precise enough time duration of the targeted process executed periodically. This makes it possible to act as if a trigger would indicate a common location in all the process segments. This virtual trigger can then be used to segment CPs from a trace. The method aims at helping researchers to reduce the effort in target preparation and in the collecting phase of the attack, while also giving a small step toward a more realistic attack scenario.

The paper is organized as follows. Section II provides the context of this work and Section III discusses related works. Then, Section V details the proposed virtual trigger segmentation method, and Section VI evaluates it experimentally on a screaming channel setup. Section VII discusses the limitation of the method and proposes a solution to overcome it. Finally, Section VIII concludes the paper.

II. Screaming channel attacks

Experimental results build on the attack scenario called screaming channels introduced by Camurati, et al. [1]. As illustrated in Fig. 1, screaming channels occur on mixed-signal devices where digital processing is collocated with analog Radio Frequency (RF) electronics over a single die. Side-channels originated from digital processing mix with RF signal and get amplified, modulated, and broadcast. The primary threat posed by screaming channels is the risk of transmitting secrets over long distances, i.e., *scream* them.

The screaming channel signals are very noisy. Plus, to collect them, it is necessary that the RF module transmits a legitimate signal. In the context of this paper, it is a Bluetooth signal. Between two Bluetooth transmissions, the collected signal contains *holes*. As in regular screaming channel analysis, to counterbalance these constraints, time diversity is used during the collection phase. The principle is to force the device to compute multiple encryptions with the same plaintext and key. Since the same data has been computed, their segment values should be the same, except for the noise. Averaging the segments returns a CP segment with reduced noise.

III. Related Works on segment synchronization

Synchronization is used to know to which CP operations each segment point belongs to. Otherwise, segment points corresponding to operations whose leakage values are data-correlated would be compared with other unrelated points. Therefore, it would be harder to distinguish a relationship between leakage values and data. We name these data-correlated points as Points of Interest (POIs). To synchronize segments, an alignment between them can be done using techniques like static alignment [9], longest common sequence [10], elastic alignment [11] and synchronous real-time sampling [12]. Before aligning segments, it is first necessary to locate these segments of interest in the traces. The most common technique in SCA research consists of inserting a trigger signal to start the trace measurement synchronized with the beginning of the CP. Attack setups are prepared to either have (1) the victim to create the trigger signal to inform the attacker when encryption starts, or (2) the attacker sending a trigger signal to the victim to make it start at a precise moment. This is an accepted scenario in the community to enable SCA research. But it assumes attackers have access to the victim to generate or listen to a trigger synchronization signal. SAKURA and NewAE's Chipwhisperer are widely used platforms in the community that follow this approach.

However, in many cases, using a trigger signal is impossible. For example, when a given firmware cannot be modified to add instructions that control the trigger signal. Or simply because the device used to collect traces is not capable of capturing two signals, the side-channel signal and the trigger signal, concurrently. Locating CPs in traces without using trigger signals can be done with pattern recognition techniques. For this purpose, Beckers, et al. [13] compare methods that calculate the correspondence between trace and pattern values. When this match score is over a pre-defined threshold, the corresponding part of the leakage is considered as being the location in the trace of a targeted segment. IcWaves[1] implements such pattern recognition methods.

Nevertheless, to use these methods, the attacker is supposed to already have a pattern or characterized segments having the same statistical properties as the researched segments, representative of the triggering moment. Therefore, the question of how to find this pattern remains open. To that end, Trautmann et al. [14] proposed a technique to locate AES CPs in leakage signals by searching for parts of the leakage having consecutive similar patterns corresponding to the 10 AES rounds. This method can find AES CPs in long traces also containing other CP operation leakages. Souissi et al. [15] used wavelet transforms to detect the limit of AES segments in traces and then used these segments to do pattern recognition.

In the screaming channels context, in order to locate CPs from leakage signals, the only technique reported so far in the literature, by Camurati et al. [1] and Wang et al. [16], used a frequency component trigger mechanism[2]. The method

[1]https://www.riscure.com/security-tools/hardware/icwaves.
[2]https://github.com/bolek42/rsa-sdr.

Input: Misaligned CP segments: $Segs_{in}$ ▷ Segments misaligned
Output: Aligned CP segment: Seg_{out} ▷ Segment aligned

1: $template \leftarrow$ None
2: **for** $index\ IN\ len(Segs_{in})$ **do**
3: **if** template == None **then**
4: $template \leftarrow Segs_{in}[index]$
5: **else**
6: $shift \leftarrow max(corr(Segs_{in}[index],\ template))$
7: $Segs_{in} \leftarrow ShiftValues(Segs_{in},\ shift)$
8: $template \leftarrow average(Segs_{in}[:index],\ axis = 0)$
9: **end if**
10: **end for**
11: $Seg_{out} \leftarrow average(Segs_{in},\ axis = 0)$

Fig. 2. **Algorithm:** Fine alignment between segments

continuously monitors a frequency band at which a given signal is present only at a unique instant of the targeted CPs. Segment locations should then correspond to trace locations where this signal is found.

In contrast with the proposed VT, all these methods require specific equipment other than the ones needed for the SCA itself, like a spectrum analyzer [1], [16] or a powerful GPU [14]. Therefore, before using more complex, expensive, or time-consuming solutions, VT is a trace segmentation method that can be used when the targeted device runs CPs without interruptions, in order to reduce the collecting phase complexity.

IV. Considered attack scenario

We consider a passive attack on system confidentiality exploiting power or EM side-channels from a CP manipulating AES keys. This causes leakage propagation through side-channels with very low Signal to Noise Ratio (SNR). The attacker capabilities include: they (1) know the precise CP duration (or can use the method presented later in the paper to find it) and (2) have access to a copy of the victim device used to build a profiled attack.

When collecting segments of interest, either during the profiling or the attacking phase, the following steps retrieve N segments from a collected trace and average them to generate a final segment with reduced noise:

1) The victim starts running a series of encryptions using the same plaintext and key.
2) The attacker, using a Software Defined Radio (SDR), starts collecting the trace after a certain delay to make sure CP executions have begun.
3) The collected trace is cut into N segments. To make sure segmentation happens only while CPs are present, N is chosen in a way that $N \times$*CP duration* is inferior to the CPs series execution duration. Except for the noise, these segments carry the same information, as each CP computed the same data.
4) A fine alignment is done between these segments, and the N aligned segments are averaged together, yielding a single representative CP segment with reduced noise (see Algorithm in Fig. 2).

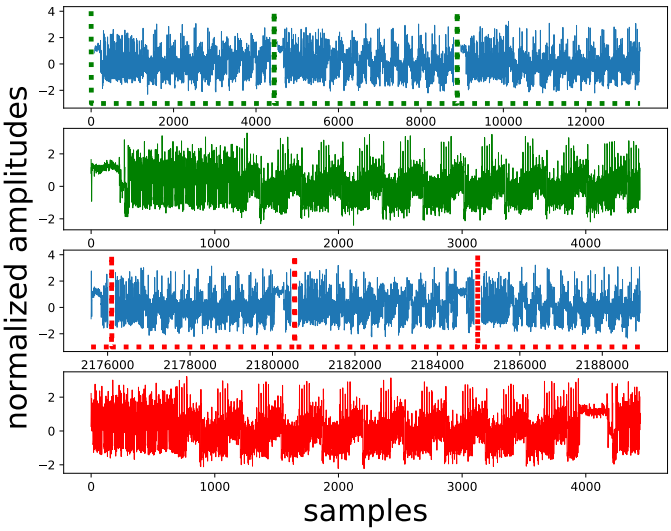

Fig. 3. **Segment CPs using the virtual trigger:** *1st row:* the beginning of the collected trace and the VT positions determined according to the known CP length. *2nd row:* a segment aligned to segment start time. *3rd/4th rows:* if the length is not precise enough, a drift appears in start times, and hence VT indicates a very different point in CPs at the end of the trace.

V. Proposed Virtual triggering method

Virtual Triggering (VT) aims at cutting each captured trace in segments (step 3 of trace collection, Section IV), with a distance between cuts being closely enough to the CP length L_{cp} to correctly segment CPs in the trace. If L_{cp} is not known very precisely, as illustrated in Fig. 3, an offset appears in the VT position. Consequently, segment starting points would be triggered at different instants in the CPs, making the final segments too different from each other to be aligned. This would deteriorate the denoised CP segment, as averaged samples would not correspond to the same instructions.

A. Find accurate CP length L_{cp}

We propose a two-step method to find the accurate CP length. This step can be performed only one time as L_{cp} is the same for all CP segments. First, an approximate value $\hat{L}_{cp}$ is found, reducing the effort to find the accurate length L_{cp} in the second step.

The first step consists in exploiting one trace containing multiple CPs. Auto-correlating such a trace, a peak is obtained each time CPs are aligned together. Therefore, $\hat{L}_{cp}$ can be approximated by manually measuring the average distance between two peaks.

The second step aims at finding the precise length L_{cp} of one CP, which is equal to the approximate length plus one δ. The method used to find the value of this δ is presented in Algorithm in Fig. 4. It consists of testing a range Δ of $\hat{\delta}$ candidates. It segments the trace with the corresponding length: $\hat{L}_{cp}$ + $\hat{\delta}$ (line 5), separates these segments into two equal groups, to distribute at best one group contains the even segments, the other the odd segments, averages each group

Input: Approximate CP length: $\hat{L}_{cp}$
Input: Range of δ $candidate\ values$: $I_{nterval}$
Output: Accurate CP length: L_{cp}

1: $\hat{\delta} \leftarrow -I_{nterval}/2$ ▷ Delta
2: $S \leftarrow I_{nterval}/2$ ▷ Stop
3: $\delta_{step} \leftarrow I_{nterval}/$number of steps ▷ Delta step
4: **repeat**
5: $Segs \leftarrow$ Cut the trace in N segments of length = $\hat{L}_{cp} + \hat{\delta}$
6: $Segment_A \leftarrow average(pair\ Segs)$
7: $Segment_B \leftarrow average(odd\ Segs)$
8: $Distance(\hat{\delta}) \leftarrow Segment_A - Segment_B$
9: $\hat{\delta} \leftarrow \hat{\delta} + \delta_{step}$
10: **until** $\hat{\delta} >$ S
11: $L_{cp} \leftarrow \hat{L}_{cp} + \hat{\delta}$ for which $Distance(\hat{\delta})$ has its lowest value

Fig. 4. **Algorithm:** How to find accurate CP length L_{cp}

to get two different reference segments A and B (line 6 - 7). Then, the L1 distance (line 8) between the two reference segments should be at its minimum when the most accurate $\hat{\delta}$ value is used. This distance is computed using equation (1).

$$L1\ Distance = \frac{abs(Segment\ A\ -\ Segment\ B)}{NB\ samples\ in\ one\ segment} \quad (1)$$

where abs is the absolute value, and *NB samples in one segment* is the number of samples in one segment.

B. Align the segments together

After having found the accurate length L_{cp} of a single CP execution, it is possible to segment correctly any trace containing the same CPs executed recursively. All the obtained denoised segments contain leakage samples from the same instructions but are not aligned together since segmentation begins at a random point in each trace. To have aligned segments, it is important to make them begin at a common point in the CP. For that, it is possible to take one of the segments as a reference and shift the values of the others until they reach the position where they best correspond to the reference segment. Afterward, a fine alignment can get them perfectly aligned to each other.

VI. Experiments

A. Experimental setup

Fig. 5 shows the setup. It comprises a victim board, a mixed-signal Nordic Semiconductor chip as in the seminal screaming channels work [1] (PCA10040[3]), processing software-based AES encryptions. On the attacker side, an SDR captures the victim RF emissions: a USRP N210[4] with an SBX daughterboard covering a frequency band from 400MHz to 4.4GHz. Instructions are sent to the victim to initialize the new plaintext value and launch a series of $N = 500$ encryptions. The signal captured by the SDR is received by the attacker's computer, which runs the pre-processing and decoding process.

[3]https://www.nordicsemi.com/Software-and-tools/Development-Kits/nRF52-DK.

[4]https://www.ettus.com/all-products/un210-kit/

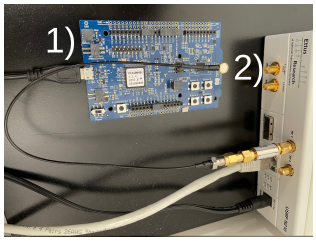

Fig. 5. **The setup**: 1) The victim PCA10040 device running AES encoding and transmitting a Bluetooth signal at 2.4GHz. 2) The SDR device collects the victim's data-correlated leakage at 2.528GHz.

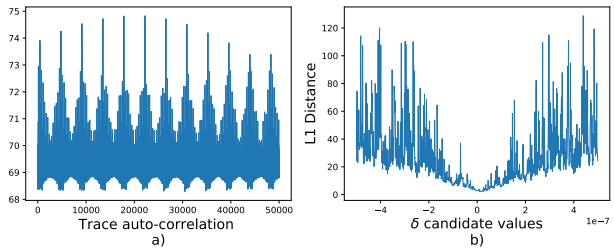

Fig. 6. **Find CP segment length**: a) Approximate length: trace auto-correlation returns a series of peaks. Each peak occurs when CPs are aligned. Manual measurement gives an approximate CP length. b) Precise length: the closer the $\hat{\delta}$ candidate value is to the real value, the more similar the two segments A and B are. Here the best $\hat{\delta}$ candidate (min. L1 distance) is 18ns.

B. Finding the accurate CP length L_{cp}

The auto-correlation results after applying the first step to find the approximate CP segment length are shown in Fig. 6 a). The average distance between two correlation peaks gives an approximate length of 4350 samples, which considering a sample rate of 5MHz, gives an approximate CP length of 870us.

The second step is performed to find a more accurate CP length. The algorithm in Fig. 4 runs on a range of 1000 ns (from -500 ns to 500 ns) with a step of 1ns. When the difference between the two CP appears to be at its minimum, this means $\hat{\delta}$ is close to the real δ value. As shown on Fig. 6 b), the best value is found at 18ns. Automating this step is kept for future work. It would consist in automatically detecting the $\hat{\delta}$ value having the lowest result. For that, it would be possible first to detect the range containing the lower minima.

C. Segmentation and alignment

Knowing the precise CP length, the trace is segmented (first row of Fig. 7 shows one segment), and the segments are finely aligned together and averaged (second row of Fig. 7). In the denoised segment, a specific instant of CPs is localized, and the segment values are shifted to make the segment begin at this instant (third row of Fig. 7). In this case, the time between two encryptions is easily recognizable as it does not vary. By computing the variance with a window of its size (230 samples), we can detect it as this variance has its lowest result at the same CP location in all segments. If the segment starting point is placed in the middle of this instant, it is then undetectable since the 230 samples are cut into two parts, one

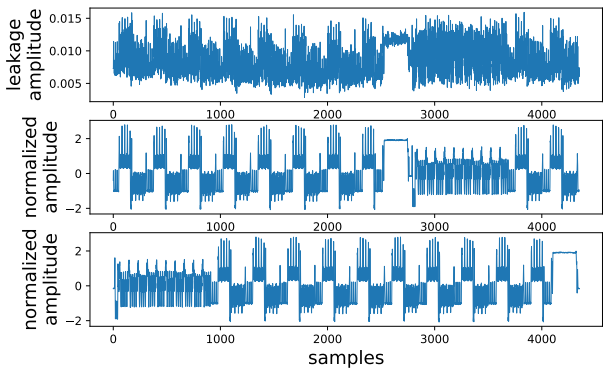

Fig. 7. **Obtain denoised and align CP segments:** *1st row:* shows a segment in the initial collected trace. *2nd row:* corresponds to averaged segments from trace cut with a precise length. *3rd row:* is the average segment aligned to a recognizable point, to be aligned with all other denoised segments. The ten rounds of AES are visible.

at the beginning of the segment, and the second at the end. To anticipate these cases, the operation is performed on the denoised segment concatenated with its own 230 first samples. Therefore, this instant can be reconstituted at the end of the segment and be detectable.

With this final step, a coarse-grained alignment is done between all denoised segments. A fine alignment (Algorithm in Fig. 2) can be necessary to have them even better aligned.

D. Comparison with original screaming channel attack

To evaluate VT, different experiments are run to compare against the original method [1]. Denoised CP segments are extracted from the same collected traces using both the VT and the frequency component triggers. 5000 segments are used to build a profile, and 200 for the attack phase. We evaluate attack efficiency with the Partial Guessing Entropy (PGE) of each AES key byte. The PGE corresponds to the rank of the correct key byte value in a list of the possible hypothesis on data from the decoding process.

Fig. 8 plots the average PGE over 30 attacks of each of the 16 key bytes sorted from best to worst, versus the number of traces used in the attack phase. The darker the blue, the lower the PGE is, i.e., the closer the attack is to guess the correct key byte. If PGE is equal to 0, the hypothesis made on the key byte is correct; if it is equal to n, then n hypothesis have a higher probability than the correct one. The red color corresponds to a PGE higher or equal to 4, which is the PGE value that no more than one key byte should reach for the key being easily recoverable in a few seconds using brute force. We can see that the PGEs are equivalent for both methods: 100 traces are enough in most cases to have only one byte with a PGE superior or equal to 4. These results show that VT, a simpler and less expensive method, achieves an equivalent efficiency to the original attack [1].

We also evaluate VT according to the CP length precision. The top left result in Fig. 9 shows the attack result after a segmentation with the most precise length we were able to determine (precision: $1ps$). The attack was repeated forcing

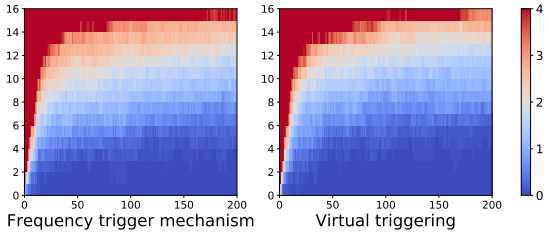

Fig. 8. **Comparison between FC and VT triggers** as PGE per key byte (the higher the blue array, the better, 16 is a maximum): the proposed method does not decrease the attack efficiency for screaming channel attacks while reducing the segmentation complexity.

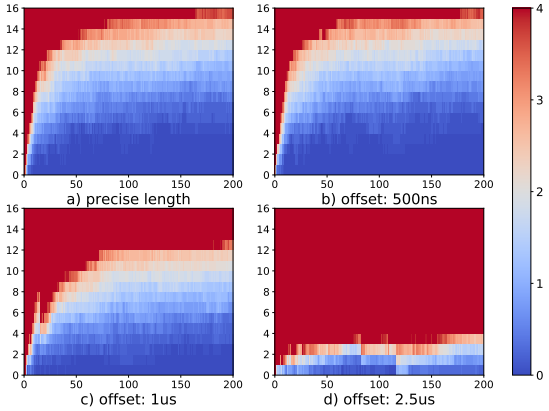

Fig. 9. **Attack efficiency for different CP length precisions**: The top left is the collection with a known length and $1ps$ precision. Other results are collected with an offset of respectively $500ns$, $1\mu s$ and $2.5\mu s$.

an offset of respectively $500ns$, $1\mu s$, and $2.5\mu s$. It can be seen that an offset of $500ns$ does not deteriorate segment information, but from $1\mu s$ the attack efficiency decreases.

E. Generalization of the method

To test the generalization of VT beyond the screaming channels context, hence on regular power/EM traces, we apply it to a signal collected by other authors [14], using leakage from a MCU device (STM32F303) executing a secure boot process containing 500 AES encryptions. As one of the requirements of our method is to have a signal containing only a series of CPs, we kept only the part of their signal containing AES segments. Using VT, the denoised segment in Fig. 10 is obtained.

VII. Improving VT with pattern recognition

VT enables pinpointing reliable points from CPs by precisely computing segments length, hence acting as if a trigger signal was used. In turn, this makes it possible to segment traces in a way that all final segments are aligned with each other. However, when delays vary between the starting instants of segments, or if the attacker targets a device executing CPs asynchronously, the method does not hold.

In these contexts, VT can still be used during a profiling phase to build a template of the CP segments for later use during an attack. Next, we propose a simple technique that

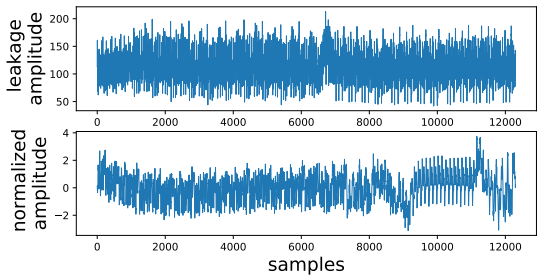

Fig. 10. **Virual trigger applied for segmentation of 500 AES of a secure boot process:**. First row: initial trace. Second row: Denoised Segment.

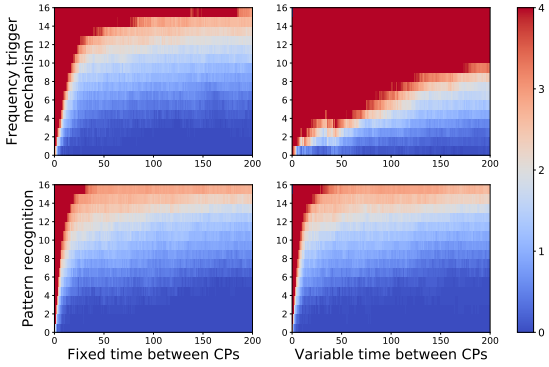

Fig. 11. **Comparison between FC trigger and pattern recognition:** Top figures show that the original screaming channels method does not resist to time variations between CPs. Bottom figures show how VT and automatic pattern pullout is resistant to time variations between CPs.

builds on VT to: (1) locate and extract a denoised CP segment from a profiling device (a copy of the victim), and learn a *segment template*; (2) automatically locate and *pullout* CP segments from traces collected on a victim instance of the device. We also demonstrate how, interestingly, this technique can absorb arbitrary time variations between CPs and improve the efficiency of existing attack methods.

During profiling, the attacker can force the device, e.g., with resets, to repeatedly execute a program containing CPs without time variations. Afterward, *during the attack phase*, the learned segment template (or only a subpart of it corresponding to a CP) can be used to locate CP segments in the traces using pattern recognition techniques. *For pattern recognition*, we cross-correlate traces with the learned pattern and detect peak positions indicating CP locations.

We compare our results with the efficiency of the FC trigger. The experiment has been performed in two scenarios: one using periodic encryptions and another with a variable delay between them. Fig. 11 shows how using pattern recognition significantly increases the attack efficiency. As previously with FC trigger, around 100 segments are needed. Using pattern recognition with the VT extracted pattern, less than 50 segments are enough.

An obvious advantage of VT compared to the pattern pullout technique with pattern recognition is that VT needs less computation effort, hence the collection phase is faster. In our case, collecting 10K denoised segments took 20 hours using pattern recognition, when only 5 hours were needed with VT. However, using the pattern pullout technique enables attacks in scenarios with time variations between CPs, which can come from system interrupts or simple countermeasures using time randomization. Nonetheless, we have not verified the method in these conditions, this is left for future work.

VIII. Conclusion

This paper has proposed VT, a simple-to-implement method to collect denoised CP segments, exploiting time diversity in the context of screaming channels. The efficiency of VT is demonstrated with respect to the state-of-the-art screaming channel trace collection method [1]. Compared to more complex and recent methods to find CPs in traces containing other operations than those exploited by the attack [14], VT is much simpler. While our experimental results concentrate on screaming channels, VT and the pattern pullout technique are generic and can be used for other forms of physical leakages.

References

[1] G. Camurati, A. Francillon, and F.-X. Standaert, "Understanding Screaming Channels: From a Detailed Analysis to Improved Attacks," *Transactions on Cryptographic Hardware and Embedded Systems*, 2020.

[2] F.-X. Standaert, "Introduction to Side-Channel Attacks," in *Secure Integrated Circuits and Systems*, 2010.

[3] J. Choi, H.-Y. Yang, and D.-H. Cho, "TEMPEST Comeback: A Realistic Audio Eavesdropping Threat on Mixed-signal SoCs," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020.

[4] P. Kocher and J. Ja, "Differential Power Analysis," *Advances in Cryptology — CRYPTO' 99*, 1999.

[5] E. Brier, C. Clavier, and F. Olivier, "Correlation Power Analysis with a Leakage Model," in *Cryptographic Hardware and Embedded Systems - CHES 2004*, 2004.

[6] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel, "Mutual Information Analysis," in *Cryptographic Hardware and Embedded Systems – CHES 2008*, 2008.

[7] S. Chari, J. R. Rao, and P. Rohatgi, "Template Attacks," in *Cryptographic Hardware and Embedded Systems - CHES 2002*, 2003.

[8] L. Masure, C. Dumas, and E. Prouff, "A Comprehensive Study of Deep Learning for Side-Channel Analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019.

[9] S. Mangard, E. Oswald, and T. Popp, "Attacks on Hiding," in *Power Analysis Attacks: Revealing the Secrets of Smart Cards*, 2007.

[10] A. Jia, W. Yang, and G. Zhang, "Side Channel Leakage Alignment Based on Longest Common Subsequence," in *2020 IEEE 14th International Conference on Big Data Science and Engineering (BigDataSE)*, 2020.

[11] K. M. Abdellatif, "Towards Efficient Alignment for Electromagnetic Side Channel Attacks," in *2019 31st International Conference on Microelectronics (ICM)*, 2019.

[12] H. Yang, E.-G. Jung, and C. Kim, "Synchronous Real-Time Sampling Technique for Side-Channel Analysis Against Randomly Varying Clock-Based Countermeasures," *IEEE Access*, 2021.

[13] A. Beckers, J. Balasch, B. Gierlichs, and I. Verbauwhede, "Design and Implementation of a Waveform-Matching Based Triggering System," in *Constructive Side-Channel Analysis and Secure Design*, 2016.

[14] J. Trautmann, A. Beckers, L. Wouters, S. Wildermann, I. Verbauwhede, and J. Teich, "Semi-Automatic Locating of Cryptographic Operations in Side-Channel Traces," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022.

[15] Y. Souissi, M. A. E. Aabid, N. Debande, S. Guilley, and J.-L. Danger, "Novel Applications of Wavelet Transforms based Side-Channel Analysis," in *Proceedings of the Non-Invasive Attack Testing Workshop*, 2011.

[16] R. Wang, H. Wang, and E. Dubrova, "Far Field EM Side-Channel Attack on AES Using Deep Learning," in *Proceedings of the 4th ACM Workshop on Attacks and Solutions in Hardware Security*, 2020.